Opportunistic Content-Centric Data Transmission During Short Network Contacts

Carlos Anastasiades, Tobias Schmid, Jürg Weber, Torsten Braun
Institute of Computer Science and Applied Mathematics
University of Bern,
3012 Bern, Switzerland,
{anastasiades, braun}@iam.unibe.ch

***Abstract*—In this paper, we investigate content-centric data transmission in the context of short opportunistic contacts and base our work on an existing content-centric networking architecture. In case of short interconnection times, file transfers may not be completed and the received information is discarded. Caches in content-centric networks are used for short-term storage and do not guarantee persistence. We implemented a mechanism to extend caching on persistent storage enabling the completion of disrupted content transfers. The mechanisms have been implemented in the CCNx framework and have been evaluated on wireless mesh nodes. Our evaluations using multicast and unicast communication show that the implementation can support content transfers in opportunistic environments without significant processing and storing overhead.**

***Index Terms*—Content-Centric, Opportunistic, Content Transmission, Mesh Nodes**

I. Introduction

Opportunistic networking, a subset of delay-tolerant networking, defines communication in challenged networks where connectivity and contact durations between devices are unpredictable and intermittent. The main goal is to exploit contact opportunities between users to support best-effort content and service interactions where fixed network infrastructure may not be available. Based on exchanged beacon messages, users detect neighboring devices as communication opportunities and need to connect to neighbors individually to perform content discovery and file transmissions.

In content-centric networks, routing and forwarding is based on content names instead of host identifiers. Nodes can express Interests to receive corresponding Data from any node in response. The exchanged messages do not contain any source or destination node addresses enabling caching in any node. Because content availability may be independent of neighboring devices, content-centric networking can support opportunistic communication without device discovery. Content discovery is performed using multicast to quickly detect available content sources. If a multicast Interest is not answered by a neighboring node, no matching content is available, which - in terms of content retrieval - is equivalent to the unavailability of neighboring devices. Received content is cached locally but persistence is not guaranteed for a long time and, therefore, it can not be used in delay-tolerant networking.

In this paper, we focus on the Content-Centric Networking approach proposed in [1], which is referred as CCN hereafter. We investigate content-centric data transmission that follows content discovery [2] in the context of short opportunistic network contacts. We implemented an extension for persistent delay-tolerant caching and evaluate it using both multicast and unicast communication.

The remainder of this paper is organized as follows: Related work is reviewed in Section II. Section III describes the required extensions for delay-tolerant content-centric Data transmission. Evaluation results are shown in Section IV. Finally, in Section V we conclude our work.

II. Related Work

A. Content-Centric Networking

CCN communication is based on two basic messages: *Interest* and *Data*. Content is organized in segments similar to chunks in BitTorrent [3]. File transfer is pull-based, and thus, users have to express Interests in every segment to obtain the entire content. CCNx [4] provides an open source reference implementation of CCN. The core element of the implementation is the CCN daemon (CCND), which performs message processing and forwarding decisions. Links from a CCND to other CCND entities are called *faces* and are defined by TCP/IP or UDP/IP sockets to other mobile hosts or by Unix sockets to local applications on the same host. A CCND has the following three main memory components:

1) The Forwarding Information Base (FIB) contains forwarding entries to direct Interests towards potential content sources.
2) The Pending Interest Table (PIT) stores unsatisfied forwarded Interests together with the face on which they were received. If Data is received in return, it can be forwarded based on face information in the PIT.
3) The Content Store (CS) is used as short-term cache in a CCN router storing received Data packets temporarily.

The *freshnessSeconds* value in the content header specifies the availability of a segment in the CS after its reception. The *Interest Lifetime* value in the Interest header determines how long an Interest remains in the PIT. Since no Interests are forwarded for existing PIT entries, the Interest lifetime has a direct impact on when Interests can be re-expressed. Data files are persistently stored and shared with others in repositories.

B. CCN Content Names

Content names in CCN [1] follow a hierarchical structure composed of arbitrary numbers and names of components. Names may follow a DNS-like structure using routable name prefixes to support Internet traffic in the form /<*domain*>/<*prefix*>/. The actual content may be stored at multiple static sites and loop-free forwarding will guarantee that no duplicates are forwarded. In case of global source mobility, naming structures using proxy-based resolution [5] and $location/identity$ splits [6], [7], [8] are proposed. By using temporal name components in the form /<*Point_of_Attachment*>/<*Device_Id*>/<*prefix*>/ content that is generated and located at specific devices can be moved together with the device to other locations. The locator *Point_of_Attachment* is a prefix that defines the location, where the device is attached. These approaches are based on a static, structured core network such as the Internet and cannot solely rely on opportunistic contacts. In opportunistic networks, the network topology is unpredictable and dynamic. In such an environment, the location of content can not be clearly specified. Based on opportunistic content exchange, content can be stored on multiple devices. A user may look for specific content names relative to the publisher's name space, e.g., */publisherA/video/* or */publisherA/audio/*. Content consumers may easily learn the naming schemes of their favorite publishers such as BBC, iTunes or Netflix. Reliable content publishers can be identified with the help of social structures built by e.g., trusted communities, reputation or rating systems.

C. Mobile Opportunistic Communication

CCN in mobile networks has already been the subject of several studies. Early works investigated the applicability of existing MANET routing protocols for mobile CCN based on analytical models [9]. A hierarchical CCN routing scheme based on distributed meta information has also been implemented [10]. The Listen First, Broadcast Later (LFBL) [11] algorithm limits forwarding of Interests at every node based on its relative distance to the content source. However, all these works assume continuous network connectivity and do not consider intermittent connectivity. Opportunistic and delay-tolerant communication has been investigated extensively in the last decade. The Bundle Protocol [12] describes a delay-tolerant protocol stack to support intermittent connectivity. Nodes can register in endpoint identifiers to form multicast trees and these registrations are exchanged when two devices meet. Content is transmitted in bursts and stored locally until the next forwarding opportunity arises. Haggle [13] describes a data-centric network architecture for opportunistic networks. The platform uses device discovery to establish point-to-point connections between nodes. Data is described by meta data composed of multiple keywords. Users express and forward interests containing keywords when connected to other devices. All data objects that match the keywords are exchanged and forwarded to the requesting node by a push-based dissemination model.

CCN can support opportunistic networking without device discovery because data transmissions are based on available content names in the vicinity. Investigations [14] already identified the potential of CCN for delay-tolerant networking (DTN). The effectiveness of CCN for opportunistic one-hop content discovery has been investigated in earlier work [2]. There are also related efforts in creating a new content-centric opportunistic networking architecture inspired by CCN [15]. In this work, we describe a cache extension to enable delay-tolerant content-centric communication and evaluate our design in a prototype implementation on wireless mesh nodes. To the best of our knowledge, this is the first work that evaluates the feasibility of CCNx [4] for DTN communication.

III. Content Transmission

During short opportunistic contacts to content sources, data transmissions may not be completed. If no alternative content sources are available, content is kept in the requester's cache until it can be completed and properly stored. Unfortunately, persistence of data in CCN caches is not guaranteed since caches are limited in size and can be overwritten by other files depending on the cache replacement strategy. Caches are built upon high-speed memory to support quick forwarding. In delay-tolerant networking, memory speed is not important since delays between successive requests are large. Therefore, in case of disruptions, partial data files can be stored on and loaded from secondary storage on the node.

A. Integration with existing solutions

Every CCN Interest is removed from the PIT if no answer is received within the lifetime of the Interest. Increasing the Interest lifetime to obtain long-living Interests may enable the integration with existing DTN protocols such as the Bundle protocol [12] but it would result in two major drawbacks:

1) Multiple Interests are required to obtain all file segments. Since a requester does not know the length of the requested file until receiving the last segment, proactive transmission of multiple Interests would be required. If all entries are valid for a long time, the PIT size would increase drastically degrading lookup performance.
2) Long-living Interests stay in the PIT and prevent forwarding of similar Interests because the request is already pending. Forwarding and retransmission is blocked for the entire lifetime period even if the environment has changed due to mobility and the content would be available.

Therefore, the Interest lifetime should be limited to a rather small value but Interests can be re-expressed periodically to account for changes in availability. As we will see later in Section IV, short Interest lifetime values are particularly advantageous during multicast communication due to shorter re-expression times in case of collisions.

B. Storage Persistence

Every content segment is named individually using a segment number. A CCN requester requests the first segment

followed by $n-1$ subsequent segments depending on the pipeline size n, i.e., the maximum number of segments that can be requested concurrently. In case of disruptions, content transfers are aborted and can be restarted again at a later time. If disruptions are short, the downloaded segments may still be available in the local cache of the requester and no redundant Interests or Data need to be transmitted. However, if disruptions are long, received objects may be removed from the cache. To obtain storage persistence, the requester needs to store the partial file and Meta Data on a secondary storage.

In the following two Subsections we will explain what *Meta Data* is required and describe the mechanisms to resume disrupted file transfers by a sample *Download Sequence*.

1) Meta Data: For every incomplete and aborted file transmission, the received partial data is stored in the file *name.part* and the Meta Data in the file *name.meta*. The required Meta Data to perform a resume operation is listed in Table I.

1. Name of content object
2. Version of content object
3. Next segment
4. File Position
5. Publisher's public key digest
6. Expiration Time

TABLE I: Meta Data stored during incomplete transmissions

Name and version of the content object can be stored together in a string. The name is used to relate the file *name.meta* to the corresponding partial file *name.part*. The third field defines the segment number that needs to be received next. The file position depends on the size of the received segments and defines where in *name.part* the new segment needs to be appended. The publisher's public key digest is used to check that the resumed file transfer is requesting content from the same publisher. To avoid incomplete files that never get completed or storing Meta Data of real-time traffic, the expiration time indicates a timeout value after which *name.meta* and *name.part* can be deleted. The expiration time is based on the *reception time* and *freshnessSeconds* of the first received segment. In case of real-time traffic, e.g., if the content is only valid for a few seconds, no Meta Data needs to be stored.

2) Download Sequence: We illustrate the content transfer with an example. Figure 1a shows a sample time sequence and Figure 1b depicts the corresponding storage management at the requester.

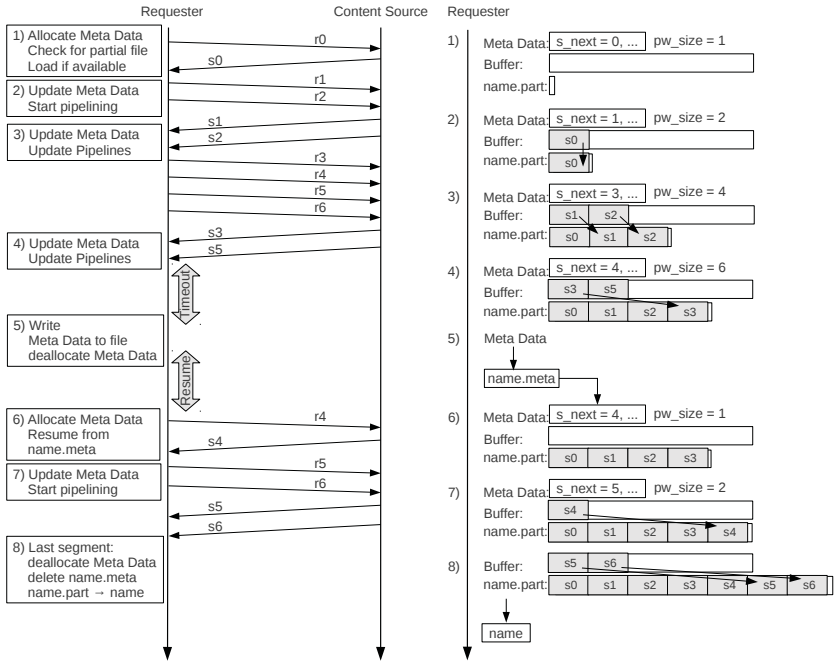
At the beginning of a file transfer in step 1), the application checks for available *name.meta* files. If *name.meta* is available, it is loaded, otherwise, the file transfer starts from the beginning by transmitting an Interest request *r0* in segment *s0*. If a content object is available and the segment is received, the requester can start expressing multiple requests at the same time in step 2). Similar to TCP slow start, the number of Interests concurrently transmitted is increasing exponentially by doubling the pipeline window size pw_{size} for correctly received segments up to the maximum value p_{max}, i.e., the pipeline size. Since requesters do not know the size of a requested content object until receiving the last segment, all Data segments are requested sequentially. The file transfer application uses a buffer of size p_{max} to temporarily store received segments until they can sequentially be written to *name.part*. In steps 3) and 4) more segments are received and pw_{size} is adapted accordingly.

In case of an Interest timeout, i.e., not receiving the segment before a timer expires, pw_{size} is reduced to 1 and the corresponding segment is requested again. This strategy targets particularly situations where requesters are disconnected from nodes that provide content and every Interest retransmission would result in a timeout. In case of timeouts due to collisions, other segments may have been received correctly and can be quickly retrieved from the CCND cache without transmitting new Interests. If the number of unsuccessful Interest retransmissions exceeds a threshold t_{thresh}, a timeout event, i.e., disconnection from the content source is assumed. In this case as shown in step 5), the Meta Data is stored in *name.meta* and all buffers are released. Therefore, segments that have already been received but not yet written to *name.part* such as segment *s5* are discarded because segment *s4* has not been received. The reasoning behind discarding is the following. First, segment sizes may vary, and thus, the length of a potential placeholder in *name.part* is not known. Second, pipelining works efficiently if it is only increased from a certain value. In case of placeholders and holes in *name.part*, more state information is required. Because the file size is not known, no fixed bit fields indicating received and missing segments similar to BitTorrent [3] can be used. However, the number of segments that are not stored due to disruptions is limited by the pipeline size p_{max}. In case of a pipeline size of 16 and a segment size of 4096 bytes, this corresponds to less than 70KB redundant data.

Detecting the availability of a content object can be performed by discovery mechanisms, which are outside the scope of this paper. In the simplest way, the environment can be probed periodically for the content. If a resume operation is performed (step 6), Meta Data is loaded from *name.meta* and the download is resumed from the last missing segment. pw_{size} starts again at 1 and is increased exponentially for correctly received segments (step 7). If the last segment has been received (step 8), which is indicated by a flag, the file transfer is finished. All Meta Data and buffers are released, *name.meta* is deleted, and *name.part* is renamed to *name*.

IV. Evaluation

We implemented content retrieval with resume capability as CCN application based on the *Meta Data* described in Subsection III-B1 and included it in the CCNx v0.6 source code. The implementation was tested on PCEngines ALIX 3D2 system boards [16] running with ADAM [17], a small-footprint embedded operating system. The current image was using Linux kernel 3.2.18. The ALIX boards have a 500MHz AMD Geode GPU, 256MB DDR DRAM and are equipped with two 802.11 miniPCI radio cards. All nodes use the IEEE802.11a wireless standard for transmission and the wireless network interfaces are configured in ad hoc mode.



(a) Resume Sequence (b) Resume Storage

Fig. 1: Download Sequence with Resume capability.

A. Evaluation Scenarios

The evaluation is performed in a static setting of two nodes: one content source shares content via a repository and one requester transmits Interests for the content. In our scenarios, we assume short contacts between requester and content source so that the entire file cannot be completed at once. We implement this by defining *disruption points*, i.e., specific numbers of segments, after which the requester will stop requesting segments emulating a disruption timeout.

In our evaluations, there is always exactly one disruption per measurement and we measure the *effective transfer time* based on two file transfers: First, the transfer time until the disruption point is reached and second, the transfer time of a second transfer that is always successful. If the resume operation is enabled, the application loads the stored Meta Data before starting the second transfer. After long disruptions, the content from the first incomplete file transfer is not available anymore in the cache. This is enforced by restarting the CCND daemon after the first transfer to clear the cache.

The total transfer time in opportunistic networks would also depend on the time a connection is disrupted, i.e. the disruption time. However, the disruption time is an additive constant that could be added to the measured effective transfer time.

In dynamic environments where neighbors change frequently, no static unicast FIB entries can be configured. Since CCN messages do not include a destination node address, they can be efficiently transmitted on wireless broadcast media using multicast MAC frames. In the following sections, we compare multicast with unicast communication by configuring the FIB accordingly. The two-node scenario indicates the baseline performance of multicast communication since there is no benefit compared to unicast. However, multicast performance will improve compared to unicast if more requesters are available at the same time. In our setting, the main differences between unicast and multicast are the mechanisms on the MAC layer. Since MAC acknowledgements are only transmitted in case of unicast, automatic retransmission can only be performed during unicast communication. Thus, the contention window, which controls the delay until a packet is transmitted on the MAC layer, can only be adapted during unicast. During multicast, the contention window is by default larger resulting in lower transfer rates.

B. Interest Lifetime for Multicast Content Transmission

During multicast communication, collisions can only be detected by unanswered Interest requests. The Interest lifetime has direct impact on Interest retransmissions and, thus, on transfer rates, because no Interests are forwarded in case of existing PIT entries. In this subsection, we evaluate different Interest lifetime values for multicast communication.

In Figure 2, we evaluate the throughput of a 2MB file with a segment size of 4096 bytes using Interest lifetimes of 4.0, 1.0, 0.5 and 0.25 seconds. The x-axis shows different pipeline sizes and the y-axis the achieved throughput. An Interest lifetime of 4 seconds is the default value in current CCNx implementations. Transmissions with large Interest lifetimes result in low data rates for small pipeline sizes because of large retransmission delays in case of packet collisions. The throughput increases with larger pipeline sizes from 16 up

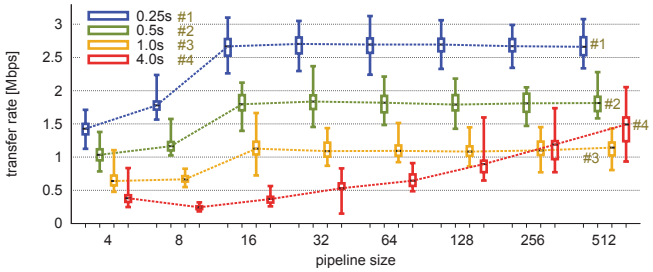


Fig. 2: Multicast Transfer rate with different Interest Lifetimes

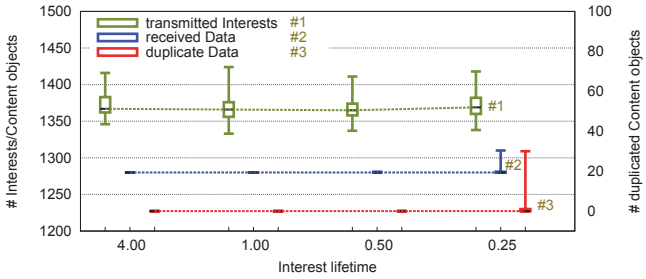


Fig. 3: Interests, Content objects and duplicate Content objects for different Interest lifetimes

to 512 by a factor of 4, i.e., from 0.38Mbps to 1.50Mbps. The reason for this increase is the larger number of Interests that are transmitted concurrently until a timeout has been detected. These Interests may retrieve and pre-fetch content that is then stored in the content store of the requester since not all of them will result in a collision. After a timeout is detected, only Interests that timed out need to be retransmitted and subsequent Interests can be served from the pre-fetched content in the cache. However, this strategy will cause 75% more transmitted Interests compared to a pipeline size of 16 due to more collisions.

While Interest lifetimes of 4 seconds are reasonable in multi-hop networks, lower values can be used during opportunistic one-hop communication. By decreasing the Interest lifetime to 0.25s, the throughput increases drastically by a factor of 7.2. Pipeline sizes above 16 and Interest lifetimes shorter than 0.25s do not result in any performance gain.

In Figure 3 we investigate the message overhead during the transmission of a 5MB file using a pipeline size of 16. The x-axis denotes the different Interest lifetimes. The left y-axis shows the number of transmitted Interests and received content objects. The right y-axis shows the number of received duplicated content objects. The number of transmitted Interests depends on the number of collisions and is the same for all evaluated Interest lifetimes. An Interest lifetime of 0.25s results in a few occasional duplicates: the median value is 0, the 75-quartile is 1 and the maximum value, which occured only once in 100 measurements, is 30. The reason for the duplicates are Interests that are retransmitted shortly before the corresponding content object is received. Since content sources do not memorize recently transmitted content, they will respond to retransmitted Interests again as we have already observed in [2]. In the worst case, an Interest lifetime of 0.25s results in an overhead of 2.34% duplicate content objects, which corresponds to less than 140KB. However, the median transfer speed increases from 0.5s to 0.25s by 48% from 1.79Mbps to 2.66Mbps.

C. Effect of Resume Capability

In this subsection, we evaluate file transfers with resume functionality as described in Section III-B2 and compare it to regular CCNx file transfer applications without resume functionality. Figure 4 shows the effective transfer time of

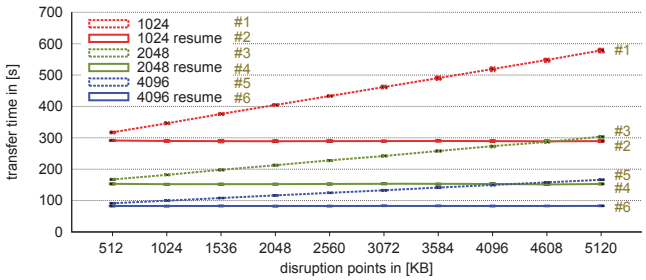


Fig. 4: Transfer time of a multicast file download with different segment sizes and disruption points

a 5MB file exchanged via multicast using different segment sizes. The colors represent the different segment sizes denoting the effective data portion in Data messages, i.e., without CCN headers containing names, signatures etc. The x-axis denotes the disruption points represented by the received KBs before the disruption and the y-axis shows the transfer time in seconds. The file transfer application with resume (continuous lines) and without resume (dotted lines) uses a stop-and-wait strategy to transmit Interests, i.e., using a pipeline size of 1.

Without resume functionality, the effective transfer times increase if the transfer is interrupted later because all segments need to be requested again in the second transfer. If the first download is interrupted immediately before the transfer has been finished, almost twice the amount of data needs to be transmitted requiring almost twice the amount of time. If resume operations are enabled, the transfer time is constant independent of the disruption points since received content is persistently stored at the requester. In our evaluations, there is always only one disruption and the second transfer attempt is always successful. Therefore, the transfer time can be reduced by up to 100%. However, in the worst case, when nodes only meet for a short time and file transfers can not be completed at once, file transfers without resume would never be completed.

The processing overhead for handling Meta Data is negligible. The size of the Meta Data depends on the size of the content name and the length of the publisher's public key digest, but it is usually significantly lower than 1KB. Resume operations do not have a negative impact on file transfers without disruptions. Evaluations of complete file

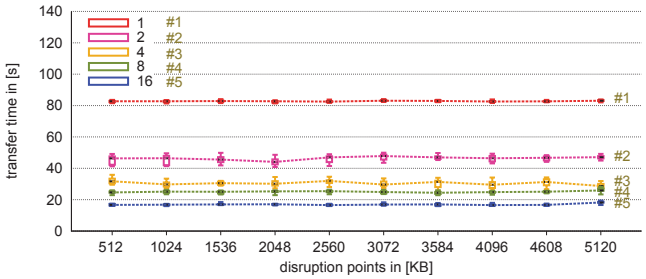


Fig. 5: Transfer time of a multicast file download with different pipeline sizes and disruption points

transfers did not show any increase in transfer time caused by Meta Data processing. The MTU of the network cards on the ALIX boards is 2274 bytes, thus, block sizes larger than 1024 result in packet fragmentation on the IP layer. We observe that despite fragmentation, larger block sizes result in shorter transfer times due to smaller data and processing overhead since fewer packets need to be transmitted.

In Figure 5 the effective multicast transfer times of a 5MB file using the resume application, a segment size of 4096 bytes and pipeline sizes of 1, 2, 4, 8 and 16 are shown. The throughput increases for pipeline sizes of 1 to 16 by 390%. The largest relative increase, i.e., 78% is observed when increasing the pipeline size from 1 to 2 because collisions do not affect subsequent transmissions in the same restrictive way. When using a pipeline size of 1, no communication is performed in case of a collision until a retransmission is performed. Due to space constraints, we do not show the results of the unicast measurements. However, the throughput increase compared to multicast is between 56% with a pipeline size of 1 and up to 97% with a pipeline size of 16. Therefore, if only one requester is available and the content source is known after discovery, direct node addressing would be favorable.

V. Conclusions

In opportunistic networks, where contacts between devices are unpredictable, CCN content discovery is performed using multicast to quickly find available content objects without knowing and configuring connections to neighboring hosts. In this work, we have shown that CCN can be used for delay-tolerant networking. Requesters need to periodically transmit multicast Interest requests to find available content sources. If connectivity to a content source is intermittent and short, file transmissions may not be completed at once and need to be resumed. Since CCN caches are only used for short term-storage, persistence is not guaranteed. Therefore, we extended caching on mobile nodes by long-term storage that persists in case of long disruptions. The required changes can be implemented as application without modification of the CCND. File transfers are resumed by maintaining Meta Data of received partial files, which is stored after disruptions. While strategies without resume operations may never be successful, resumed file transmissions result in constant effective transfer times independent of the disconnection time from a content source. Evaluations showed that the processing and storage overhead is negligible and does not affect content transfers in any way.

If content sources are unknown, transfers need to be performed by multicast. In the absence of MAC layer acknowledgments in case of multicast communication, collisions may result in long retransmission delays degrading the achievable throughput. Since opportunistic communication is performed via one hop, the Interest lifetime can be decreased to a lower value to reduce retransmission delays. Evaluations have shown that decreasing the Interest lifetime can increase multicast throughput by a factor of 7.2 without significantly increasing the number of transmitted messages.

As future work, we will evaluate the resume functionality in mobile scenarios using a hybrid emulation framework.

Acknowledgments

The work presented in this paper was partially supported by the Swiss State Secretariat for Education and Research under grant number C10.0139.

References

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, “Network Named Content,” in *5th ACM CoNEXT*, Rome, Italy, December 2009, pp. p. 1–12.

[2] C. Anastasiades, A. Uruqi, and T. Braun, “Content discovery in opportunistic content-centric networks,” in *5th IEEE WASA-NGI*, Clearwater, FL, USA, October 2012, pp. p. 1048–1056.

[3] B. Cohen, “Incentives Build Robustness in BitTorrent,” in *1st P2PEcon*, Berkely, USA, June 2003, pp. p. 1–5.

[4] (2013, September) CCNx. [Online]. Available: http://www.ccnx.org/

[5] J. Lee, D. Kim, M.-W. Jang, and B.-J. Lee, “Proxy-based mobility management scheme in mobile content centric networking (ccn) environments,” in *ICCE*, January 2011, pp. p. 595–596.

[6] F. Hermans, E. Ngai, and P. Gunningberg, “Mobile sources in an information-centric network with hierarchical names: An indirection approach,” in *7th SNCNW*, Linkping, Sweden, May 2011.

[7] R. Ravindran, S. Lo, X. Zhang, and G. Wang, “Supporting seamless mobility in named data networking,” in *IEEE FutureNet V*, 2012.

[8] D.-H. Kim, J.-H. Kim, Y.-S. Kim, H. s. Yoon, and I. Yeom, “Mobility support in content centric networks,” in *IEEE Sigcomm ICN Workshop*, Helsinki, Finland, August 2012.

[9] M. Varvello, I. Rimac, U. Lee, L. Greenwald, and V. Hilt, “On the Design of Content-Centric MANETs,” in *8th WONS*, Bardonecchia, Italy, January 2011, pp. p. 1–8.

[10] S. Y. Oh, D. Lau, and M. Gerla, “Content Centric Networking in Tactical and Emergency MANETs,” in *IFIP Wireless Days*, Venice, Italy, October 2010, pp. p. 1–5.

[11] M. Meisel, V. Pappas, and L. Zhang, “Listen First, Broadcast Later: Topology-Agnostic Forwarding under High Dynamics,” in *ACITA*, London, UK, September 2010, pp. p. 1–8.

[12] K. Scott and S. Burleigh. (2007, November) Bundle protocol specification. RFC 5050. [Online]. Available: http://tools.ietf.org/html/rfc5050

[13] J. Su, J. Scott, P. Hui, J. Crowcroft, E. D. Lara, C. Diot, A. Goel, M. H. Lim, and E. Upton, “Haggle: seamless networking for mobile applications,” in *9th UbiComp*, Innsbruck, Austria, September 2007, pp. p. 391–408.

[14] G. Tyson, J. Bigham, and E. Bodanese, “Towards an information-centric delay-tolerant network,” in *2nd IEEE NOMEN*, Turin, Italy, April 2013.

[15] B. Batista and P. Mendes, “ICON - An Information Centric Architecture for Opportunistic Networks,” in *2nd IEEE NOMEN*, Turin, Italy, April 2013.

[16] (2013, September) PCEngines. [Online]. Available: http://www.pcengines.ch/

[17] T. Staub, S. Morgenthaler, D. Balsiger, P. Goode, and T. Braun, “Adam: Administration and deployment of adhoc mesh networks,” in *3rd IEEE HotMESH*, June 2011, pp. p. 1–6.